Approximate Policy Improvement for Continuous Action Set-Point Regulation Problems

A.O. Esogbue and W.E. Hearnes II
Intelligent Systems and Controls Laboratory
Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta GA 30332-0205
e-mail: aesogbue@isye.gatech.edu and wp46268@west-point.org

Abstract— **Model-free control methods based on classical dynamic programming approximation algorithms are a fertile area of research. These methods are generally for discrete state and action spaces. This research proposes an algorithm for extending approximate policy iteration to continuous action spaces by combining the derivative-free line search methods of nonlinear optimization with policy improvement based on Q-learning using a discrete subset of reference actions. The properties of the proposed algorithm are discussed and two example problems illustrate its applicability. Q-learning algorithm are used to search a continuous action space. The method reduces the computational effort required in many problems.**

Keywords— **Dynamic programming, policy iteration, Q-learning, continuous action spaces, inverted pendulum, power system stabilization.**

I. Introduction

As real-world control problems become more complex, the use of traditional control techniques requiring mathematical models of the plant become less appealing or appropriate. The motivation for model-free intelligent control stems from the ability of humans to determine efficient and, sometimes, near-optimal control of highly complex nonlinear systems through online experience without a priori knowledge of the plant dynamics. Intelligent controllers have several important advantages that make them attractive for real-world applications. These include higher robustness, shorter development time, and less assumptions about the dynamical behavior of the plant.

Approximate methods based on dynamic programming (DP) are a fertile and ongoing area of research for model-free intelligent control. Among the major classes of algorithms are Sutton's temporal differences (TD) [**?**], Watkin's Q-learning [**?**], [**?**], and Baird's advantage updating [**?**]. These are online versions of the policy improvement and successive approximation algorithms in classical dynamic programming [**?**]. Considering reinforcement learning can be formulated as a dynamic program, a number of reinforcement learning controllers reported in the literature, including [**?**], [**?**], have approximate DP-based learning algorithms.

Dynamic programming methods, including many of the model-free algorithms, are generally based on a finite action space. Continuous state or action spaces bring on the curse of dimensionality unless some type of approximation is used. This paper proposes an algorithm for approximating the optimal control policy for set-point regulation problems with continuous action spaces. It takes advantage of the property that the optimal policy is usually found considerably quicker than the optimal functional values in the Q-learning algorithm. A derivative-free search algorithm determines an improved policy during each policy improvement phase. Convergence properties of the new algorithm are discussed and two experimental control problems show its practical applicability.

II. The Set-Point Regulation Problem

Set-point regulation problems, a subclass of terminal control problems, require a process to be driven to a desired final state, defined as the set-point, in an optimal manner according to some scalar performance measure. This class includes problems such as the inverted pendulum balancing problem and the power system stabilization problem, both of which are used as practical examples below.

We assume throughout that the system in question is controllable. With a finite number of states, the dynamics of the system can be modeled via a

Markov process. While the transition probabilities $p_{ij}(\mathbf{a})$ may not be known, they are assumed to exist. Since the process is controllable, the set-point $\mathbf{s}^\star$ is accessible from any state i. Specifically, $p^n_{i\mathbf{s}^\star} > 0$ for some $n \geq 0$.

For clarity, we define the model of the set-point regulation problem examined in this research. The process to be controlled is described by a discrete-time Markov chain P with discrete state space $\mathbf{S}$ and continuous action space $\mathbf{A}$, where $\mathbf{S} = S$. Let the unknown dynamics of the system be expressed as

$$\mathbf{s}(k+1) = \tau(\mathbf{s}(k), \mathbf{a}(k), w(k)) \tag{1}$$

for $k = 0, 1, \ldots$ where $\mathbf{s}(k) \in \mathbf{S}$ is the state, $\mathbf{a}(k) \in \mathbf{A}$ is the action, and $w(k)$ is the random disturbance at time k. Define the probability that $\mathbf{s}(k+1) = j$ when in $\mathbf{s}(k) = i$ and action $\mathbf{a}(k)$ is taken to be $p_{ij}(\mathbf{a}(k))$. Furthermore, let the immediate return $R : \mathbf{S} \times \mathbf{A} \to \mathbb{R}$ for taking action $\mathbf{a}(k)$ in state $\mathbf{s}(k)$ be bounded below by 0 and above by some integer B.

For any given control policy $\pi : \mathbf{S} \to \mathbf{A}$, define this objective function as

$$V_\pi(i) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(\mathbf{s}(k), \mathbf{a}(k)) | \mathbf{s}_0 = i \right] \quad \forall i \tag{2}$$

and let

$$V(i) = \inf_\pi V_\pi(i) \quad \forall i \tag{3}$$

where $\gamma \in (0, 1)$ is a discount factor and E_π is the conditional expectation using policy π. $V_\pi(i)$ represents the discounted expected total return using policy π and starting in state i. The objective function is defined over an infinite horizon since the set-point regulation problem is a terminal control problem where the number of stages until the set-point is reached is not fixed. Rather, it is dependent upon the policy π. Therefore, with an appropriate boundary condition of $V(\mathbf{s}^\star) = 0$, the terminal control problem is appropriately formulated. Formulating the relation as a dynamic program, the functional equation becomes

$$V(i) = \min_{\mathbf{a} \in \mathbf{A}} \left[R(i, \mathbf{a}) + \gamma \sum_{j \in \mathbf{S}} p_{ij}(\mathbf{a}) V(j) \right] \tag{4}$$

for all i which represents the minimum expected discounted return when starting in state i and always following an optimal policy.

III. Policy Iteration Algorithms

Howard's classical policy improvement algorithm [**?**] involves solving a system of linear equations for the unknown values of $V_\pi(i)$. Using these values, a new policy π' is determined that is a strict improvement for at least one state i. The convergence of the algorithm is a direct result of the property that an optimal stationary policy exists and there can be only a finite number of stationary policies.

In practice, the transition probabilities $p_{ij}(\mathbf{a})$ required for Howard's algorithm may not be known. Subsequently, model-free versions have been developed. In addition, if the action space is not finite, as in the case of a continuous action space, then the convergence of the algorithm is not guaranteed.

Bradtke, et. al, derive some of the first convergence results for adaptive policy iteration with continuous action spaces [**?**]. The linear-quadratic regulator (LQR) problem is well known in control theory. The structure of the problem leads to some elegant analytical results. Bradtke, et. al, exploit some of the characteristics of the LQR problem to derive an analytical policy improvement procedure. Although the number of stationary policies is infinite, the algorithm converges to the optimal policy. Unfortunately, these analytical results apply only to LQR problems. In this research, we are interested in a broader class of control problems.

IV. Online Policy Iteration Via Q-Learning

Dynamic programming solves two related fundamental problems: (1) Determination of the optimal functional value $V(i)$ for each state i, and (2) a policy $\pi^\star$ that achieves that value. In many control problems, the optimal *policy* is the more important part of the solution. The value of the functional equation when using an optimal policy is not necessarily required and, in many cases, online dynamic programming algorithms such as Q-learning expend additional computational effort to determine these values.

Watkin's Q-learning algorithm is an online version of the classical successive approximations algorithm in DP [**?**]. The Q-values in the Q-learning algorithm can be defined so that they either learn the functional values for the optimal policy $\pi^\star$ with

$$Q(i, \mathbf{a}_j) = R(i, \mathbf{a}_j) + \gamma \sum_{l \in \mathbf{S}} p_{il}(\mathbf{a}_j) V(l) \tag{5}$$

with

$$V(i) = \min_j Q(i, \mathbf{a}_j) \tag{6}$$

or for a fixed policy π with

$$Q(i, \mathbf{a}_j) = R(i, \mathbf{a}_j) + \gamma \sum_{l \in \mathbf{S}} p_{il}(\mathbf{a}_j) \sum_h \xi_l^\pi(\mathbf{a}_h) Q(l, \mathbf{a}_h) \quad (7)$$

where $\xi_l^\pi(\mathbf{a}_h)$ is the probability of choosing action $\mathbf{a}_h$ in state l under policy π. Equation 7 reduces to Equation 5 under the optimal stationary policy $\pi^\star$.

Just as there is a fundamental relationship between the optimal policy and the optimal functional values in DP, there is also a fundamental relationship between policy improvement and successive approximation algorithms. Bertsekas and Tsitsiklis [**?**] highlight two parameterized algorithms, λ-policy and modified policy iteration, that show this relationship.

This research proposes a type of uniform grid search to approximate the $Q(i, \mathbf{a}_j)$ function for each state i and reduce the interval of uncertainty containing the minimum of the functional. The line search problem in nonlinear optimization is the foundation of a number of optimization algorithms [Bazaara, et al]. If the $Q(\cdot, \mathbf{a}_j)$ functions were strictly quasi-convex then a number of efficient derivative-free line search methods could be used with guaranteed convergence. The $Q(i, \mathbf{a}_j)$ functions, however, are not necessarily strictly quasi-convex, therefore a variation of the uniform search method is used.

V. CAS Algorithm

The continuous action space (CAS) algorithm begins with a representative subset $\mathbf{a}_j, j = 1, \ldots, A$, for each state i, spanning some interval of uncertainty (IoU) regarding the location of the optimal control action in the continuous action space. The Q-learning algorithm determines the optimal *policy* using only this action subset. Based on this policy, the interval of uncertainty is reduced for selected states, thereby adjusting the locations of the reference actions. As the CAS algorithm continues, the intervals of uncertainty for each state are reduced toward 0, centering on the optimal action in the continuous action space, if certain assumptions are maintained.

The general CAS algorithm is as follows:

1 Set boundary condition: $V(\mathbf{s}^\star) = 0$.
2 Initialize $\mathbf{a}_j$ for all states.
3 Set $Q_0(i, \mathbf{a}_j) = M \gg 0 \quad \forall i, j$.
4 $Count := 0$
5 **while** $Count < Limit$ **do**
6 Perform an iteration of Q-learning.
7 **if** Policy doesn't change
8 $Count := Count + 1$
9 **else**
10 $Count := 0$
11 **fi** **od**
12 Reduce IoU by $\beta < 1$ around $\pi(i) \quad \forall i$
13 Repeat steps 3-12 until desired accuracy.

The choice of β, the threshold $Limit$, and the choice of M all affect the rate of convergence. The properties are examined in the next section.

VI. Properties of CAS Algorithm

The key to the efficiency of the CAS algorithm is that the optimal policy can, in many cases, be determined before the Q-learning algorithm converges to the optimal functional values. Basing the policy improvement procedure on this information is equivalent to waiting for the Q-learning algorithm to converge.

The Q-values from Equation 5 are equivalent to a positive stochastic dynamic program and, therefore, an optimal stationary policy exists [**?**]. A stationary policy is one that is nonrandomized and is time-invariant. Our set-point regulation problem is controllable and has an absorbing state that is accessible by every other state, therefore, the estimates will converge with probability 1: $Q_k(i, \mathbf{a}_j) \to Q(i, \mathbf{a}_j)$.

Proposition .1: Given a Markov system P with an absorbing state and a unique optimal stationary policy, there exists an $\epsilon > 0$ sufficiently small such that if the Q-learning algorithm converges to the optimal values in $k \in \mathbb{N}$ iterations then the optimal policy is determined in $k' \leq k$ iterations.

The proof of Proposition .1 is omitted for brevity. It provides only a weak theoretical upper bound on the number of iterations until the optimal policy is found. In practice, though, the optimal policy may be found in significantly less iterations. For example, consider a discrete approximation to the inverted pendulum balancing problem. With the objective to minimize the discounted sum of the squared error from the set-point, the dynamic programming functional equation becomes

$$V(i) = \min_{\mathbf{a}_j} \left[(\theta^2 + x^2) + \gamma \sum_{l \in \mathbf{S}} p_{il}(\mathbf{a}_j) V(l) \right]. \quad (8)$$

The state space consists of $51 \times 5 \times 51 \times 5$ discrete states for θ, $\Delta\theta$, x, and Δx, respectively. Each $Q(\mathbf{s}, \mathbf{a})$ value is arbitrarily initialized at some large integer. Using a full backup for each iteration, 3401 iterations are necessary before the Q-values converge to within $\epsilon = 0.0001$ of the optimal functional

values. Yet, the algorithm determines the optimal policy after only 36 iterations, as shown in Figure 1 which plots these policy changes during the 3401 iterations.

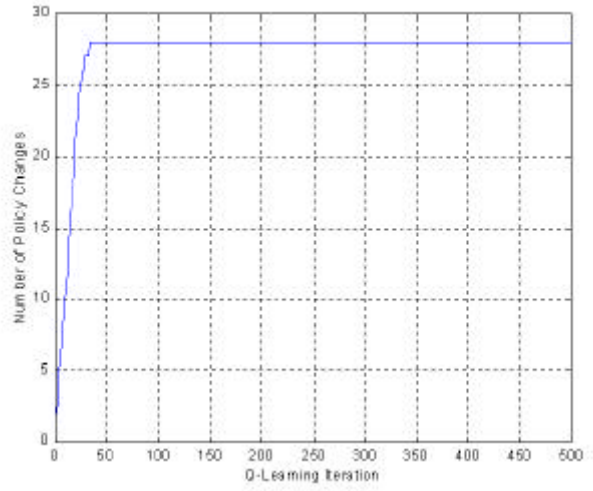

Fig. 1. Optimal policy found in considerably less iterations than the optimal functional values in an inverted pendulum balancing example.

The estimates of the $Q(i, \mathbf{a}_j)$ values for each state i serve as a guide for reducing the interval of uncertainty. Initially, this interval is the entire action space $\mathbf{A}$. Each reduction is by a factor of β, $0 < \beta < 1$, and therefore the interval can be made arbitrarily small using successive reductions. The procedure that reduces the interval of uncertainty for each state i is as follows:

1 Determine $\text{argmin}_j Q_k(i, \mathbf{a}_j) \quad \forall i$.
2 Reduce current IoU by β for each i.
3 Center IoU on action in step 1.
4 Spread A reference actions uniformly in IoU.

Proposition .2: Apply the Q-learning algorithm to estimate the $Q(i, \mathbf{a}_j)$ values. From Proposition .1, we have two possibly distinct times k' and k. The successive reduction procedure above produces identical reductions in the interval of uncertainty at both k' and k.

Clearly, the successive reduction procedure is based on $\text{argmin}_j Q_{k'}(i, \mathbf{a}_j)$ and $\text{argmin}_j Q_k(i, \mathbf{a}_j)$, respectively, $\forall i$. From Proposition .1, it follows that

$$\text{argmin}_j Q_{k'}(i, \mathbf{a}_j) = \text{argmin}_j Q_k(i, \mathbf{a}_j), \quad \forall i.$$

Therefore, the procedure produces identical reductions in the IoU at both k' and k.

In order to use the successive reduction procedure for guaranteed policy improvement, the $Q(i, \mathbf{a}_j)$ functions must be strictly quasi-convex in j. Specifically, this is equivalent to a type of topological nearness in the $Q(i, \mathbf{a}_j)$ for all i such that if the optimal action is $a^\star$ then

$$\|\mathbf{a}_l - a^\star\| < \|\mathbf{a}_j - a^\star\|$$

implies

$$|Q(i, \mathbf{a}_l) - Q(i, a^\star)| < |Q(i, \mathbf{a}_j) - Q(i, a^\star)|$$

for $j \neq l$. This ensures that if some action is optimal, then actions close to that optimal action are better than actions further away.

Proposition .3: If the $Q(i, \mathbf{a}_j)$ function is strictly quasi-convex in j for all i then the successive reduction procedure generates a set $\mathbf{a}'_j$ of A reference actions for i such that

$$\min_j Q(i, \mathbf{a}'_j) \leq \min_j Q(i, \mathbf{a}_j).$$

The proof of this follows directly from the strictly quasi-convex property of the Q-function and is omitted here. As stated previously, this assumption does not always hold but, even then, an adroit choice of β and A can achieve the desired result in practice. This is evident in the example applications that follow.

The CAS algorithm converges to the optimal policy $\pi^\star$ under the strictly quasi-convex assumption if the successive reductions are applied to a particular sequence of states.

Proposition .4: If the $Q(i, \mathbf{a}_j)$ function is strictly quasi-convex in j for all i then the successive reduction procedure can determine the optimal policy $\pi^\star$ for all states.

Again for brevity only, the outline of the proof is given. The state space $\mathbf{S}$ can be divided into subsets according to the expected number of actions in the optimal control policy required to move the process to the set-point. If the states that have only one expected transition to the set-point have their IoU's reduced first, then their optimal actions will be found. By induction and Bellman's principle of optimality, a similar case can be made for states that are $2, 3, \ldots, z$ expected transitions away from the set-point under an optimal policy. This gives a theoretical convergence, but in practice, this division is not known a priori. Heuristic procedures are thus used to approximate this partition.

VII. Experimental Application

The CAS-C algorithm is applied to two set-point regulation applications. First, as a testbed control problem, the inverted pendulum balancing problem

is examined. Second, the power system stabilization problem, an underdetermined system, is investigated. Both are multiple-input/single-output nonlinear control problems that are approximated by a fine discrete state space.

A. Inverted Pendulum Balancing

This control problem has four inputs: [θ, $\Delta\theta$, x, Δx] and one output F. The objective is to balance the pendulum at $\theta = 0°$ and keep the cart at $x = 0$. With the functional equation defined as in Equation 8 and the state space discretized into $51 \times 7 \times 51 \times 7$ states, the CAS-C algorithm was run.

For a standard to compare the CAS-C algorithm, the Q-learning algorithm was run with 101 discrete actions to give a fine approximation to a continuous action space. This requires 12,872,349 Q-values. After 105 full-backup iterations, the Q-learning algorithm converged to within $\epsilon = 0.00001$ of the optimal functional values for each state-action pair. This required over 1,351,596,645 Q-value updates. With the CAS-C algorithm, only $A = 7$ actions were used with a reduction factor $\beta = 0.8$. The CAS-C required 437 full-backup iterations but, with the reduced action space, it only required 389,866,491 Q-value updates. This is a reduction of over 70% in the computational effort.

The differences in the optimal trajectories between the ideal and the CAS-C approximation are shown in Figures 2 and 3. Since the strictly quasi-convex assumption may not hold, we are not guaranteed convergence. However, a choice of a higher β may increase our probability of finding the optimal action but at the price of increased computational effort.

B. Power System Stabilization

The power system stabilization problems has six state variables, only two of which can be measured. Therefore the system is only partially observable. In this experiment, the control system consists of two inputs: [ω, $\Delta\omega$] and one output u. The stabilization system is composed of a synchronous machine with an exciter and a stabilizer connected to an infinite bus. The dynamics of the synchronous machine (used for simulation only) are represented by a linearized incremental model [**?**]. The objective is to dampen the oscillations [ω, $\Delta\omega$] under various load changes. With the functional equation defined

$$V(i) = \min_{\mathbf{a}_j} \left[\omega^2 + \gamma \sum_{l \in \mathbf{S}} p_{il}(\mathbf{a}_j) V(l) \right]. \qquad (9)$$

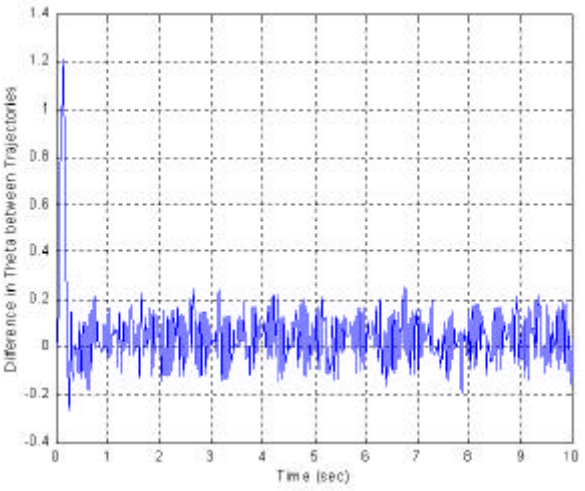

Fig. 2. Error in optimal trajectory of θ over time for initial state [10,0].

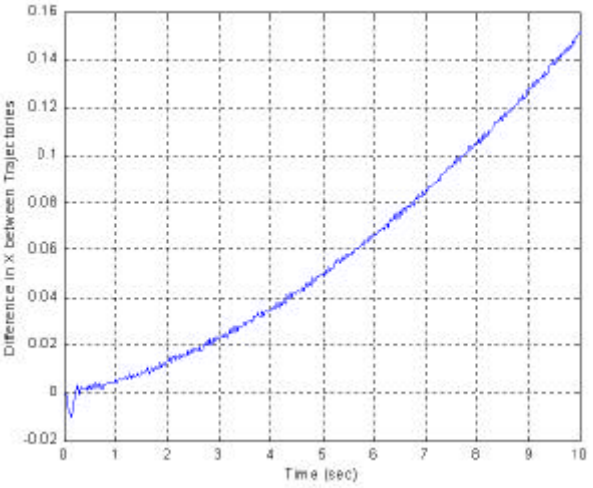

Fig. 3. Error in optimal trajectory of x over time for initial state [10,0].

and the state space discretized into 101×51 states, the CAS-C algorithm was run.

The inverted pendulum example above was run using full backups, which required knowledge of the state transitions. This was done for demonstration purposes. In the power system stabilization problem, sample backups are used to learn a model-free control law. Because the system is only partially observable with the four unobservable state variables taking on independent values, comparisons to an optimal policy cannot be made. The number of reference control actions were $A = 7$ with $\beta = 0.8$ and 16 reductions of the IoU were per-

formed. The stabilization of the system after the completion of learning when presented with a load change of $0.05pu$ is shown in Figure 4. The control sequence is shown in Figure 5.

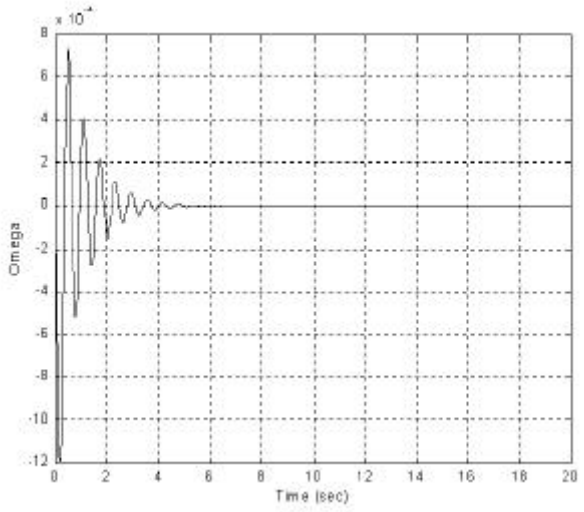

Fig. 4. Trajectory of ω over time for initial state [0.01,0].

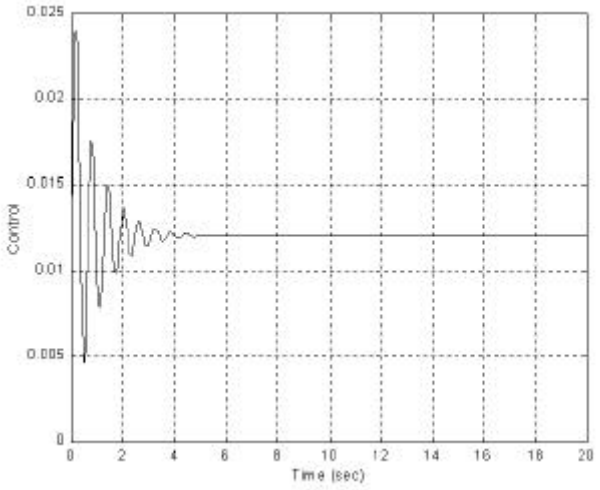

Fig. 5. Learned control policy of u over time for initial state [0.01,0].

Additional research into the sample backup approach and its effect on the number of Q-value updates, β, and A is underway in our laboratory.

VIII. Conclusions

The CAS-C algorithm combines the derivative-free line search methods of nonlinear optimization with online dynamic programming to create an approximate policy iteration procedure to estimate an optimal control law. The convergence properties of the Q-learning algorithm are used to search a continuous action space using a discrete subset of reference actions. The reduction in computational effort over a fine discrete approximation is evident. Example applications are shown for both full and sample backups.

Under the assumption of strict quasi-convexity, the policy is indeed improved in each successive reduction step. If the reductions in the intervals of uncertainty are applied systematically, convergence to the optimal control action is assured. In practical applications, where the assumption may not hold, an adroit choice of β and A can increase the probability of finding the optimal action.

Research is currently underway in our laboratory into heuristics for ensuring a proper sequence of interval of uncertainty reductions, a non-uniformly distributed set of reference actions, and a fuzzy approximation scheme for extending this to continuous state spaces.